The Lattice QCD Workflow Provenance Framework

Luciano Piccoli James Kowakolski James Simone Xian-He Sun

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, IL, USA 60510
lqcd-workflow@fnal.gov

Abstract

Lattice Quantum Chromodynamics (LQCD) workflows produce large amounts of data resulting from numerical simulations of QCD quantum field theory. Configuration files generated through Monte Carlo simulations are later used for computing certain physics quantities such as decay rates and particle masses. Workflow systems can be used to help standardize workflow descriptions, but lack tracking of domain specific information. Additionally, queries on provenance metadata require domain scientists to learn specific languages. In this paper we propose a model for tracing data and execution provenance independent of workflow system. We believe the model is applicable for scientific problems in general while allowing domain specific information to be included. We use the Ruby on Rails framework to implement the data model in conjunction with the openWFEru BPM execution engine. Provenance queries can be composed using the Ruby language, which is more accessible than raw SQL statements. We show some preliminary results that indicate little performance impact generated by recording the provenance information and show some examples of data retrieval using Ruby.

1. Introduction

Unprecedented amounts of data are currently produced and analysed by e-science experiments. The organization of this massive information is critical for its effective use in new discoveries. Lattice Quantum Chromodynamics (LQCD), the numerical study of QCD quantum field theory on a four-dimensional discrete lattice, generates considerable data that are processed at several institutions. Applications, software libraries, input data and workflow recipes are shared among collaborators worldwide.

Unlike many e-science experiments, which use Grid resources for harvesting capacity processing power, LQCD computations employ tightly-coupled parallel processing which requires computers with high-speed low-latency networks. Binary codes are fine tuned to exploit capabilities of each underlying architecture. LQCD workflows can effectively exploit the capacity of one or more parallel computers by running many independent computations at once.

LQCD workflows are categorized into two typical types: configuration generation and analysis campaign. The former is used for creating an *ensemble* through Monte Carlo simulations. An ensemble is an ordered collection of gluon configurations sharing the same physics parameters (e.g. lattice spacing and masses). An analysis campaign iterates over an ensemble to compute physics quantities such as decay rates and particle masses. The processing for each configuration is independent of the other configurations. Many such analysis campaigns are conducted on each ensemble.

Provenance is one of the most important requirements for LQCD workflows [10]. It is necessary for tracing files origins, including the workflow execution (instance) and which job (participant) generated the file along with physics parameters. An important aspect of a provenance system is the ease of formulating queries which can be difficult for a domain scientist, partly because it usually involves learning an unfamiliar language.

The configuration generation and analysis campaign workflows require coordination of physics parameters, execution environment parameters, binaries, and input, output and log files. Python, Perl and shell scripts are currently used to execute workflows. Final and intermediate data files are generated as workflow participant outputs. Some secondary data products which serve mainly as data quality indicators (e.g. conjugate gradient residuals, average plaquettes and status messages) are simply printed and become scattered among the multitude of log files generated by workflow scripts. The loose organization and control of generated files, products, parameters and workflows result in difficult to answer questions such as: Where is the configuration file *l612f21b6600m0290m0484.6*? What where the input parameters used to generate *l612f21b6600m0290m0484.6*? Which configuration files where generated using algorithm su3_rmd version 1.2? What are the configuration files gen-

erated with residuals smaller than 10E-5?

Most workflow systems lack detailed provenance data and execution tracing. These features are critical for LQCD and similar large scale scientific workflows. Tracking of outputs in forms other than files (e.g. error messages, parameters and checksums) is needed for provenance to be complete. Two workflow management systems were evaluated using sample LQCD workflows and the provenance features provided were considered not sufficient [10].

Considering that generic provenance modeling is not trivial to adapt for specific problems, we developed our own model driven by concrete LQCD workflow requirements. The main contributions of this paper are in the provenance data model and its implementation. Despite targeting a specific problem, the model still allows modifications to support other scientific applications. That is possible in part due to the use of a very flexible and extensible framework for the prototype implementation.

The proposed provenance framework for LQCD is described in section 2. Section 3 shows the prototype implementation and discusses advantages of using Ruby on Rails. The prototype was evaluated in regards to the ability to perform queries by domain scientists and performance impact of the provenance framework on typical workflows (section 4). Related work is discussed on section 5. Finally, section 6 summarizes our contribution and future work.

2. LQCD Provenance Framework

In order to address the provenance needs of LQCD workflows we propose a framework that is independent of workflow systems and based on specific requirements of this field. Our goal is to provide data provenance, execution tracking and parameter management for all LQCD workflows. The framework allows tracking of: provenance of generated data; values and history of secondary products; workflow input parameters; and history of execution and environment used by participants and workflows. In addition it is possible to compose advanced queries about parameters, results, workflow history and status.

Freire [7] recognizes that provenance querying is closely tied to storage models used. In fact it is cumbersome for a scientist to compose SQL queries to retrieve responses for the sample questions listed before. In section 4 we discuss more on how to minimize use of SQL by using a very flexible language for provenance querying.

2.1. Data Model

We use an object-oriented model for describing the provenance classes and their relationships. Groups of classes are implicitly divided into four spaces: parameter, data provenance, secondary data, and process history spaces. The spaces do not define a hard boundary between sets of classes, but rather a logical and functional aggregation.

The parameter space archives all parameters used as input for workflows, including physics parameters (e.g. quark masses), algorithmic parameters (e.g. convergence criteria) and execution parameters (e.g. number of nodes used). Parameters are name-value pairs that can be grouped in sets. Groups of parameters are used to describe the physics properties of ensembles or hold analysis campaign attributes. Parameter sets are optionally identified by names. A simplified class diagram showing the parameter space components is show in figure 1.

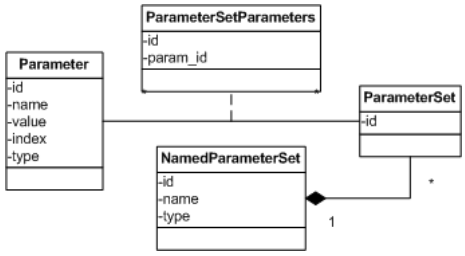

Figure 1. Parameter space

The relationship between input and output files is kept within the data provenance space (see figure 2). Data products are modeled as Products, which have optional ProductProperties. An example of LQCD product is the ensemble configuration file *l612f21b6600m0290m0484.6* that has a property named *u0* with value 0.94. Its parent configuration file is the product named *l612f21b6600m0290m0484.3* and child *l612f21b6600m0290m0484.9*. For more complex analysis campaign workflows it is possible to have multiple parent-children relationships. The products also have a reference to the workflow participant instance that generated it, allowing the file to be reproduced by reconstructing the processing steps.

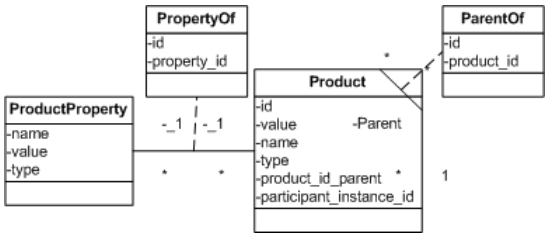

Figure 2. Data provenance space

In addition to data files and their properties, LQCD workflows also produce secondary data very specific to the experiment. Figure 3 shows the classes that hold secondary data. These classes have meaning only in the LQCD context

and would need to be expanded in case the model is used in conjuction with other scientific workflows.

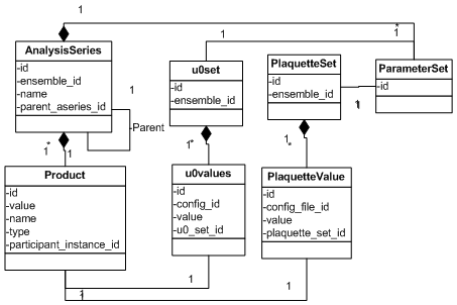

Figure 3. Secondary data space

The process history space (figure 4) holds information regarding workflows and participants. Each generic class of participant is defined as a ParticipantType (e.g. numerical integration). An actual implementation of a ParticipantType is realized by a Participant (e.g. Gauss algorithm version 2). The Participant holds information about the binary code, including command line format, version and pre and post run scripts. The actual execution of a Participant is recorded as a ParticipantInstance (e.g. Gauss algorithm version 2 ran successfully on node A producing file X).

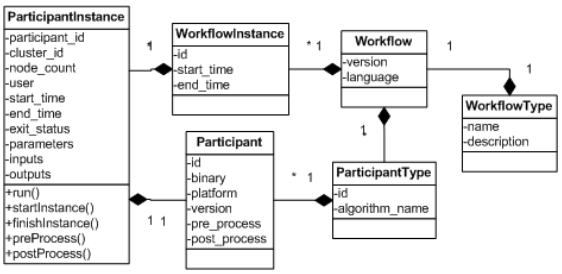

Figure 4. Process history space

Even though the provenance framework is designed to be workflow system independent, it is necessary to track the relationship between ParticipantInstances and workflows. A similar construct used for managing participants is used for tracking workflows. WorkflowType defines a class of workflow (e.g. Two point analysis). The Workflow describes the implementation of the type using a workflow language (e.g. BPEL4WS). The execution of a workflow causes the creation of a WorkflowInstance, which has references to ParticipantInstances. The analysis of the process history space in conjunction with the data provenance space allows the recreation of complete execution traces of workflows.

3. Prototype Implementation

For the implementation of the framework prototype we chose the Ruby programming language and the Ruby on Rails (RoR) framework [14]. Ruby is a flexible and dynamic object-oriented language similar to Python and Perl used by the RoR framework. It allows quick and easy development of web applications. The key feature of RoR in regards to the provenance tracking system is the implementation of the active record pattern [6].

Active record defines the connection between the classes defined in the previous subsection 2.1 and database tables. RoR offers transparent connections to a variety of databases, including Postgres and Mysql. Furthermore, the active record allows the use Ruby as provenance query language, addressing problem raised by Freire [7]. The classes' attributes are mapped to database table columns and relationships between classes are described in the model definition independently from the database.

Figure 5 is a simplified view of the prototype implementation. A web interface is the main entry point for domain scientists. It allows users to add, modify or remove parameters and parameters sets besides running and monitoring available workflows.

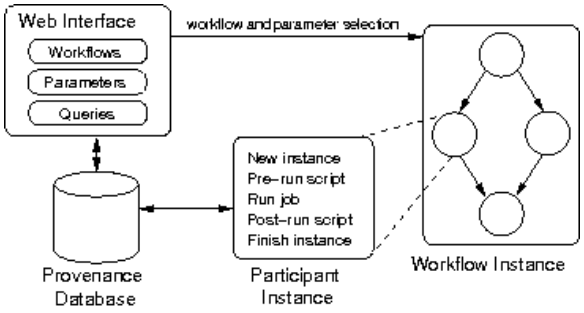

Figure 5. Prototype implementation diagram

After a workflow and parameters are selected a workflow execution engine is launched. The openWFEru BPM engine [15] is used for the prototype. It is a business oriented workflow system with good support for workflow patterns [1] implemented in Ruby, which integrates very well with the RoR framework.

As participants are invoked by the engine new ParticipantInstances are created in the database. Every participant optionally has a pre and post run script that are invoked before and after running the actual binary code. The pre-run scripts are used for translating parameters from the database parameter space into command line arguments suitable for the participant binary. After finishing the participant execution the post-run script saves provenance records. The prototype currently submits jobs to PBS/Maui running on clusters at Fermilab.

The retrieval of provenance information about current and past workflow executions is possible via the web interface through pre-packaged queries or by using Ruby syntax as shown in section 4.

Other benefit of RoR framework not currently exploited by the implemented prototype is the access of the data via web services. This feature can be used for providing a provenance service to remote workflow executions.

3.1. Workflow Systems Integration

A goal of our provenance model is to provide a workflow system independent tool. It is however a challenging task to integrate provenance with a workflow management system. Both systems are required to interface on every operation that generates provenance data, for example when a participant starts or finishes.

Although applications are the components best suited to generate the data provenance for the workflow [12] the codes may not be easily changed to include provenance tracking mechanisms. Thereby we use wrapping with pre and post run scripts, allowing legacy applications to be integrated with the system.

In the prototype, produced products are declared to the provenance system by the participant wrapper, contained within post-run script. Depending on the workflow system used, it is possible to record provenance through callbacks using events.

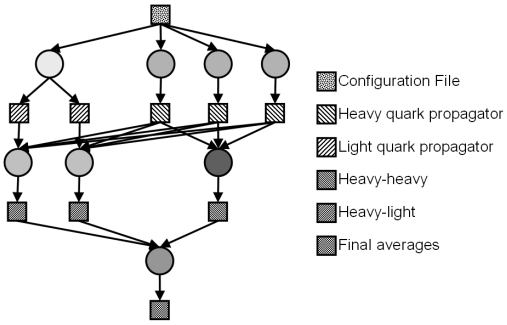

Figure 6. Two-point analysis workflow for one configuration file

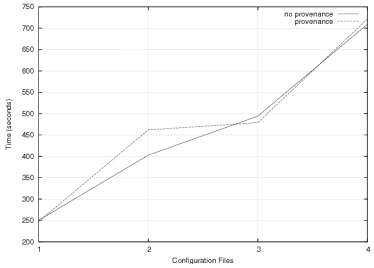

Figure 7. Provenance framework overhead for the two-point analysis campaign workflow

4. Evaluation

A two-point analysis campaign workflow was used to evaluate the overhead introduced by the provenance framework. For each file in the input ensemble a workflow similar to figure 6 is executed. The same workflows run with provenance enabled and disabled for comparison purposes. Workflow participants for this setup use only a single node, even though the participants are MPI jobs capable of running on multiple machines. Jobs are submitted a special PBS queue that has two reserved nodes in the main production cluster.

The total running time for the two-point workflow with both setups are shown on figure 7. The horizontal axis shows the number of configuration files used from the input ensemble. There is no significant impact of recording the provenance on the tested workflows. In fact the running time in many cases is lower when provenance is recorded, possibly due to delays in the cluster scheduler and related components. Figure 8 shows the same running time per participant.

One of the advantages of the provenance framework is the ability to use the Ruby language to perform specific queries based on the proposed model. Results from the queries can be manipulated within the language or exposed to scientists through web interfaces. The queries show in figure 9 provide answers to questions posed in the introduction using the interactive Ruby shell.

The format of queries follow an object-oriented style since the provenance products are instances of classes defined by the data model. This format is more intuitive than SQL statements, although some flavor of SQL is still present on the sample queries shown. A common approach to minimize direct iterations with the provenance database is to define a set of canned queries. The RoR environment can be used to quickly make these default queries accessible through the web.

5. Related Work

Provenance on e-Science is addressed by several systems [11]. Some of which focus on provenance only as PASOA [13], PASS [9] and Karma [12], or are workflow systems aware of provenance such as Pegasus [5] and Kepler [2].

PASOA defines a provenance service that records workflow traces. The traces contain detailed information about web services invoked during workflow execution. Compo-

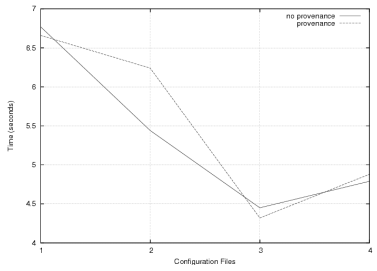

Figure 8. Provenance framework overhead per participant instance

```
$ Product.find(:all, :conditions => \
        "file_name like '%1612f21b6600m0290m0484.6%'")
...
$ su3 = ParticipantType.find(:first, \
                  :conditions => "name == 'su3'")
$ su3.participant.participant_instances.products
...
$ properties = ProductProperty.find(:all, \
                :conditions => "name == 'error'" and
                               "value > 10E-5")
$ products = properties.products
```

Figure 9. Sample provenance queries using Ruby

nents of PASOA provide valuable services such as the retrival of traces in a human readable format and validation of web service calls based on logged information of past service calls.

PASS proposes a very interesting approach for provenance tracking that is completely independent of workflow systems. It records events at the operating system level saving fine grained provenance data within the file system. The transparency of this system makes it a great tool for system administradors, but has limited applicability to LQCD workflows. PASS restricts workflows to run on supported operating systems only and limits the provenance data to translations from low-level events (e.g. command line invokations).

Karma provides a provenance framework independent of workflow systems. A publish/subscribe message passing system is used for transmitting provenance information from applications to a central server. Optionally it is possible to monitor the running workflows by subscribing to provenance messages.

Pegasus and Kepler are workflow-based systems with provenance features. Pegasus originally used the Virtual Data System (VDS) to capture provenance. Currently Pegasus uses PASOA [8] for a more complete provenance support, including tracking of transformations between abstract to concrete workflows. On the other hand, Kepler does not directly support provenance, but extensions are available [4, 3].

Our prototype system has similarities with the PASOA framework, differing on the data model and recording strategy (service vs. active record). Extending our framework to support a provenance service is an alternative for running workflows on remote sites. Such addition is possible through web services support from the RoR framework.

An advantage of the LQCD provenance framework is the flexibility to add application specific produced data into the model. Furthermore, cluster monitoring data can be added to the same database and used by the provenance framework and workflow engine to provide enhanced workflow fault tolerance features. Health information about computing nodes is already being collected at some LQCD clusters using similar provenance data model concepts.

6. Conclusions and Future Work

In this paper we presented our work on the LQCD provenance framework. The implemented prototype accomplishes the objectives of record provenance of generated data, secondary products, workflow input parameters and workflow execution history. We believed the proposed data model can also be used with other e-science problems provided that specific domain data is included in the secondary data space.

A preliminary evaluation of the prototype shows that overall workflow performance is not affected by the addition of the provenance recording mechanisms. Furthermore, provenance queries can use the full power of the Ruby language avoiding SQL intricacies.

There are several improvements additions planned to the prototype. Long lasting workflows may be spawn off to clusters at remote locations with distinct acccess policies. The system should be capable of handling database backloading with results from external processing. The development team also focuses on the integration issues with different workflows systems, tests with Kepler and Pegasus are planned.

7. Acknowledgments

This work was supported in part by Fermi National Accelerator Laboratory, operated by Fermi Research Alliance, LLC under contract No. DE-AC02-07CH11359 with the United States Department of Energy (DoE), and by DoE SciDAC program under the contract No. DOE DE-FC02-06 ER41442.

References

[1] W. M. P. V. D. Aalst, A. H. M. T. Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distrib. Parallel Databases*, 14(1):5–51, 2003.

[2] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. Kepler: an extensible system for design and execution of scientific workflows. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 423–424, 2004.

[3] S. Bowers, T. M. McPhillips, B. Ludäscher, S. Cohen, and S. B. Davidson. A model for user-oriented data provenance in pipelined scientific workflows. In *IPAW*, pages 133–147, 2006.

[4] S. M. S. da Cruz, P. M. Barros, P. M. Bisch, M. L. M. Campos, and M. Mattoso. Provenance services for distributed workflows. In *CCGRID*, pages 526–533, 2008.

[5] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Sci. Program.*, 13(3):219–237, 2005.

[6] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, and R. Stafford. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002.

[7] J. Freire, D. Koop, E. Santos, and C. T. Silva. Provenance for computational tasks: A survey. *Computing in Science and Engineering*, 10(3):11–21, 2008.

[8] S. Miles, E. Deelman, P. Groth, K. Vahi, G. Mehta, and L. Moreau. Connecting scientific data to scientific experiments with provenance. In *E-SCIENCE '07: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*, pages 179–186, Washington, DC, USA, 2007. IEEE Computer Society.

[9] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer. Provenance-aware storage systems. In *ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, pages 4–4, Berkeley, CA, USA, 2006. USENIX Association.

[10] L. Piccoli, J. B. Kowalkowski, J. N. Simone, X.-H. Sun, D. J. Holmgren, N. Seenu, A. G. Singh, and H. Jin. Lattice qcd workflows: A case study. In *SWBES '08*, Indianapolis, IN, USA, 2008.

[11] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34:31–36, 2005.

[12] Y. L. Simmhan, B. Plale, and D. Gannon. A framework for collecting provenance in data-centric scientific workflows. *icws*, 0:427–436, 2006.

[13] M. Szomszor and L. Moreau. Recording and reasoning over data provenance in web and grid services. In *International Conference on Ontologies, Databases and Applications of Semantics (ODBASE'03)*, volume 2888 of *Lecture Notes in Computer Science*, pages 603–620, Catania, Sicily, Italy, Nov. 2003.

[14] D. Thomas, D. Hansson, L. Breedt, M. Clark, J. D. Davidson, J. Gehtland, and A. Schwarz. *Agile Web Development with Rails*. Pragmatic Bookshelf, 2006.

[15] P. Wohed, B. Andersson, A. H. ter Hofstede, N. Russell, and W. M. van der Aalst. Patterns-based evaluation of open source bpm systems: The cases of jbpm, openwfe, and enhydra shark. Technical report, 2007.